MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2017/2018

DCS5088 – OBJECT ORIENTED PROGRAMMING
(For DIT students only)

28 OCTOBER 2017
9.00 am – 11.00 am
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This examination paper consists of **12** pages.

2. **SECTION A:** There are **3** structured questions.

3. **SECTION B:** There is **1** structured question.

SECTION A: Structured Questions (Total: 70 Marks)

Instruction: Please write all your answers in the Answer Booklet provided.

QUESTION 1 (30 marks)

1.1 Given the following code segments, identify the output for each of them.

a)
```
for ( int y = 20; y <= 120; y *= 2 )
        cout<< y ;
```
[3 marks]

b)
```
x = 1;
while(x <= 18)
{
        cout<< x ;
        x += 6;
}
```
[3 marks]

1.2 Given the program below:

```
#include<iostream>
using namespace std;
//1.2 a) write your answer on your answer booklet

void get_input(struct Employee&);
void taxation(struct Employee&);

int main()
{
   Employee Vee;
   get_input(Vee);
   taxation(Vee);
   cout << "\nName        : " << Vee.name
      << "\nSalary   : RM " << Vee.salary
    << "\nTaxation : RM " << Vee.tax;


}
//1.2 b) write your answer on your answer booklet
//1.2 c) write your answer on your answer booklet
```

Sample output screen

```
Enter name       : Jean Perry
Enter salary     : 5600.55

Name     : Jean Perry
Salary   : RM 5600.55
Taxation : RM 560.055
```

[Note: Refer to sample output given. The ***bold*** items are the inputs entered by user]

Continued...

a) At segment labelled *'// 1.2 a)'* , declare a *structure* named *Employee* which consists of three data members :
- *name* (***string***)
- *salary* (***float***)
- *tax* (***float***)

[2 marks]

b) At segment labelled *'//1.2 b)'*, write the function definition for the function prototype (`void get_input(struct Employee&);`). In this function, user will enter *name* and *salary* for an employee. [3 marks]

c) At segment labelled *'//1.2 c)'*, write the function definition for the function prototype (`void taxation(struct Employee&);`). In this function, the employee's tax will be determined based on the table below.

salary	*tax*
At least 10,000.00	20% of *salary*
Less than 10,000.00 but at least 5,000.00	10% of *salary*
Less than 5,000.00 but at least 3,000.00	5% of *salary*
If all the above is false	0

[8 marks]

1.3 Given the program below:

```
#include <iostream>
using namespace std;
class Furniture {
   private:
      int order;
      float cost;
   public:
      float price();
      void setCost (float c) { cost = c; }
      void setOrder(int o)    { order = o; }
      void setOrder() { order = 3; }
      float getOrder() { return order; }
}

//1.3 a) write your answer on your answer booklet

int main()
{   Furniture Cavenzi;

    //1.3 b) write your answer on your answer booklet

    cout << "The total cost is: RM " << Cavenzi.price() << endl;
    return 0;
}
```

Continued...

a) At segment labelled ***'//1.3 a)'***, define member function *price( )* outside the class. The function calculates and return *cost* × *order.* [2 marks]

b) At segment labelled ***'//1.3 b)'***, write the codes to do the following:
   i. Using object *Cavenzi*, call function *setCost(...)*, passing in float value 42.5.
   ii. Using object *Cavenzi*, call function *setOrder(...)*, passing in integer value 7.

[2 marks]

c) Trace and write the output produced once the whole program is complete. [2 marks]

d) There are **FIVE** errors when the following statements are added in the **main()** function of the same program. Correct the errors by rewriting the program statements that contains those errors. [5 marks]

```
furniture Lee;
Lee.cost = 300.05;
Lee.order = 2;
cout << "Lee qty: " << Lee.order <<
" costs : RM "  Lee.price() << endl;
```

QUESTION 2 (20 Marks)

2.1 Given the program below:

```
#include <iostream>
using namespace std;
class Table
 {  int width_measure, length_measure;
    public:
      // 2.1 a) Write your answer on your answer booklet

       void Set_Measurement (int L, int T)
      {    width_measure = L;
           length_measure = T;
      }

      friend class Building;
  } ;
```

Continued...

```
class Building
{ string name;
  public:
      Building()
      {  name = "PU9";
         cout<<"----Buiding name : "<<name<<"-------"<<endl;
      }

     // 2.1 b) Write your answer on your answer booklet

};

int main()
 {   Building B1;
     Table t[5];
     int i  = 0, x, y;
     while( i < 5 )
     { cout<<"Enter the tables' width and length :\n";
       cin>>x>>y;
       t[i]. Set_Measurement(x,y);
       i++;
     }

     B1.findLargest(t);
     return 0;
 }
```

[**Note:** Refer to sample output given below. The ***bold*** items are the inputs entered by user]

Sample output screen

```
----Buiding name : PU9-------
~~Object Created~~
~~Object Created~~
~~Object Created~~
~~Object Created~~
~~Object Created~~
Enter the tables' width and length :
4 13
Enter the tables' width and length :
2 4
Enter the tables' width and length :
5 21
Enter the tables' width and length :
21 2
Enter the tables' width and length :
```

Continued...

```
3 9
The table area :52
The table area :8
The table area :105
The table area :42
The table area :27
The largest area is 105
```

a) At segment labelled *'//2.1 a)'*, write the constructor function that outputs "~~Object Created~~". [2 marks]

b) At segment labelled *'//2.1 b)'*, write the codes to define function *findLargest(...)*. This function will receive an array of 5 *Table* objects and display the area (*width* x *length*) of each object. The largest table area will be determined and displayed. [9 marks]

2.2 Given the program below:

```
#include <iostream>
#include <cmath>
using namespace std;
class Triangle
{  protected: double a, b, c;
   public:
       Triangle(double x=3)
       {  a=x; b=3; c=3;
          cout<<"--PP1--"<<endl;
       }
       Triangle(double x, double y)
       {  a=x; b=y;
          cout<<"--PC2--"<<endl;
       }
};

class Pythagoras : protected Triangle
{    public:
        Pythagoras(double x, double y) : Triangle(x,y)
        { cout<<"--Pythagoras--"<<endl;
        }

        double find();
};

double Pythagoras::find()
{      c = sqrt(a*a + b*b);
       return c;
}
```

Continued...

```
int main()
{       Pythagoras may(3,4);
        cout<<"Hypertenuse :"<<may.find()<<endl;
}
```

a) Analyze the program above and fill in the blanks for the statements below.
   - Supposed *Pythagoras* class is inherited by *Yy* class using protected inheritance, the public member *find()* of *Pythagoras* class will be seen as _________ in the *Yy* class. [1 mark]
   - Supposed *Pythagoras* class is inherited by *Yy* class using private inheritance, the protected members (double a, b, c) of *Triangle* class will be seen as _________ in the *Yy* class. [1 mark]

b) Trace the output for the above program. [3 marks]

c) Dynamic memory allocation is not utilized at the main function. Rewrite the main function to incorporate dynamic memory allocation. *[Tip: You will need to write the codes to allocate memory and also to deallocate memory]*. [4 marks]

QUESTION 3 (20 Marks)

3.1 Given the program below:

```
#include<iostream>                                          // line 1
using namespace std;                                        // line 2
                                                            // line 3
class Coordinate                                            // line 4
{                                                           // line 5
 private:                                                   // line 6
            double a,b;                                     // line 7
 public:                                                    // line 8
      Coordinate  () { a = b = 0.0; }                       // line 9
                                                            // line 10
      Coordinate  (double a_arg, double b_arg)              // line 11
       {          a = a_arg;                                // line 12
                  b = b_arg;                                // line 13
       }                                                    // line 14
                                                            // line 15
      + operator(const Coordinate  &) {                     // line 16
      return Coordinate(a + p.a, b + p.b);                  // line 17
     }                                                      // line 18
                                                            // line 19
      void display()                                        // line 20
      {   cout<<"a :"<<a<<", b :"<<b<<endl; }               // line 21
```

Continued...

```
};                                                          // line 22
                                                            // line 23
int main()                                                  // line 24
{  Coordinate ob1, ob2(1.5, 3.5), ob3 (5.0, 7.0) ;          // line 25
   ob1 = ob2 + ob3;                                         // line 26
   ob1.display();                                           // line 27
   ob2.display();                                           // line 28
   return 0;                                                // line 29
}                                                           // line 30
```

a) Identify **TWO** errors in the class by copying the lines that have the errors and rewrite the lines with the necessary corrections. [2 marks]

b) Trace and write the output produced by the program (Note: Assuming there are no coding errors). [4 marks]

3.2 Trace and write the output produced by the program below: [7 marks]

```
#include<iostream>
using namespace std;
class Job
{    protected:
        float salary;

     public:
         virtual void display ( float s )
         {  salary = s;
            cout<< "= Your salary is :: RM " << salary<< endl;
         }

         virtual ~Job()
         { cout<< "Job class" << endl << endl; }
};

class Lecturer: public Job
{       void display( float s )
        {  salary = s;
           cout<< "This is a Lecturer's salary :" << salary << endl;
        }

        ~Lecturer()
       {  static int num=1;
          cout << "Lecturer " << num << " salary calculation done" <<
          endl ;
          num++;

        }

};
```

Continued...

```
int main()
{   Job j;

    Job *p = new Lecturer;
    p->display(1500);
    delete p;

    p = new Lecturer;
    p->display(2590);
    delete p;

    p = &j;
    p->display(150);

   return 0;
}
```

3.3 Observe the program below:

```
#include <iostream>
#include <string>
using namespace std;

class HumanBody
{
      protected:
            string name;
            float height, weight;
      public:

      // 3.3 a) Write your answer on your answer booklet


};

class BMI  : public HumanBody
{
      private:
            float bm;
      public:
            BMI(string nm, float wg, float hg)
            {
                  bm = 0.0;
                  name=nm;
                  height=hg;
                  weight=wg;
            }
            float calcBMI()
            {
                  bm = weight / (height * height);
                  return bm;
            }

            void display()
            {
```

Continued...

```
                    cout << "~Name          : " << name << endl;
                    cout << "~Height (meter) : " << height << endl;
                    cout << "~Weight (kg)   : " << weight << endl;
                    cout << "~BMI           : " << calcBMI()<<endl;
             }

};

int main()
{    HumanBody B;
     BMI obj;

     return 0;
}
```

a) At segment labelled *'//3.3a)'*, set `void display(void)` as a pure virtual function. [3 marks]

b) The program has errors at the ***main()***. Explain what is wrong with the object declarations. [4 marks]

Continued...

SECTION B (Total: 30 Marks)

Instruction: Please write all your answers in the Answer Booklet provided.

Write a **complete program** that gets input from user for the quotation of **THREE** categories of stationary. The program also will display the total quotation received and the final total price (discounted total quotation).

[**Note:** Refer to sample output given below. The ***bold items*** are inputs entered by user.]

➢ Declare a constant for the program:

- *SIZE* (int) : Set it to constant value of 5. This value is for the size of the *month* array of *Quotation* class that is used in this program.

➢ Create class called *Stationary:*

- Protected data members:
  - o *type* : string
  - o *category* : string

- Public member function:
  - o *setStationary (....)* : Contains two string parameters to set the *type* and *category.*

➢ Create class called *Quotation* [**derived publicly** from class *Stationary*]:

- Private data members
  - o *group* : string
  - o *month* : string[SIZE]
  - o *quo* : float[SIZE]
  - o *total* : float

- Public member functions:
  - o *Parameterized constructor* : - Contains three parameters of string type. One of the parameter is used to set the *group* and the other two parameters will be passed to function *setStationary (...)* to set the *type* and *category.*
    - Set the *month* array with values *"Jan", "Feb", "Mac", "Apr", "May".*
    - Set the total to 0.

Continued...

o	*setQuotation ()*	: - Get user input for *quo* array. - Accumulate the *quo* array element in *total*.
o	*display()*	: - Display the *category*, *type*, *group* and *total*.
o	*FinalPrice (Quotation&)*	: - Set this function as a *friend* of the class. Refer to the instructions after this to define the function.

➢ Create a friend function called *FinalPrice(Quotation&)*.

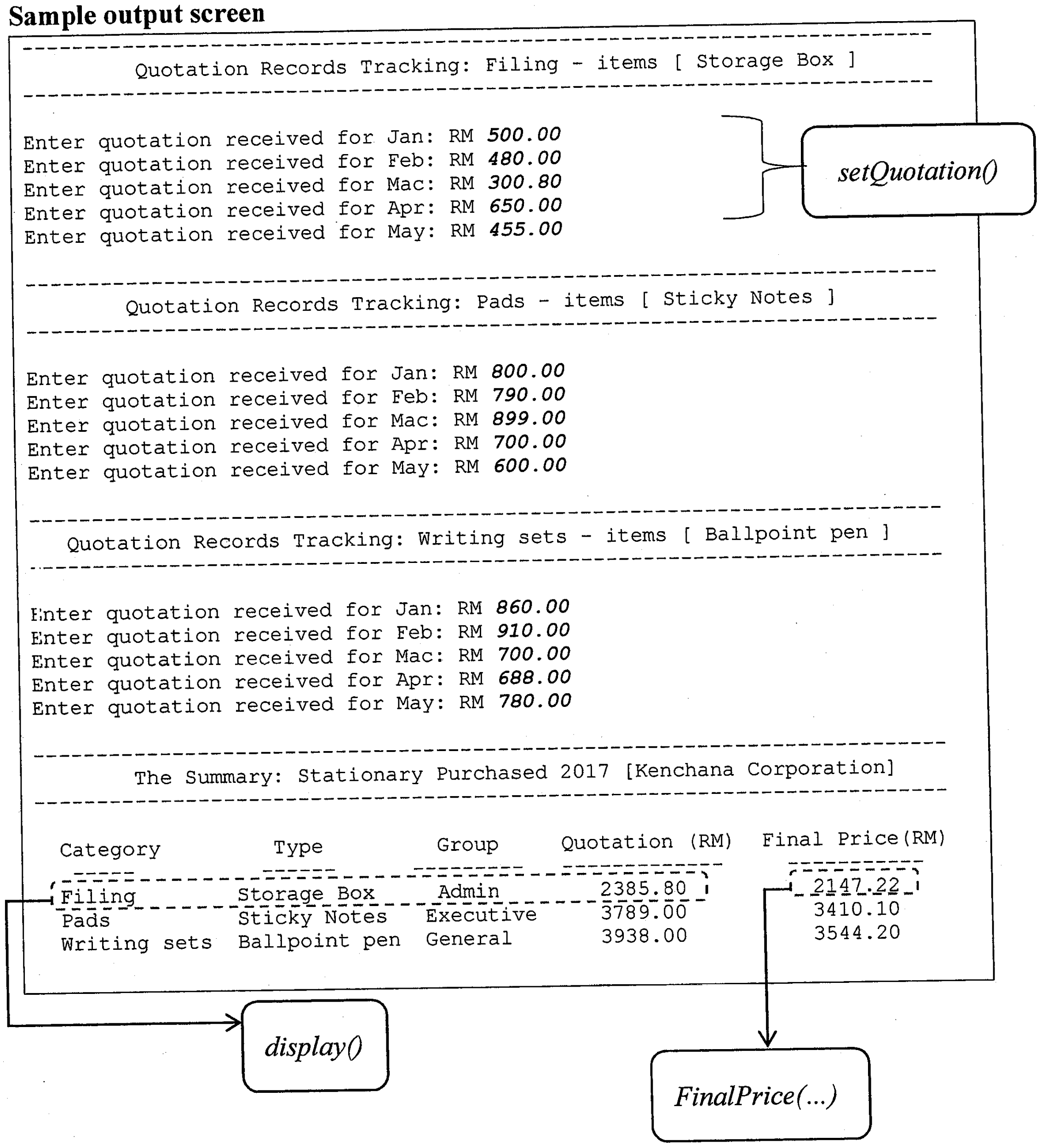
- Parameter : Reference object of *Quotation* class.
- Returns the Final Price [**Hint:** Final Price = 10% discount from the *total* quotation (use data member of *Quotation* class)].

In ***main()***:

- Declare a pointer object, ***a*** of *Quotation* class.
  - Use this pointer to create a dynamic object element. Pass the values **"Filing"**, **"Storage Box"**, **"Admin"** that will set the object's *category*, *type* and *group*.
  - Call *setQuotation ()*.
- Declare a pointer object, ***b*** of *Quotation* class.
  - Use this pointer to create a dynamic object element. Pass the values **"Pads"**, **"Sticky Notes"**, **"Executive"** that will set the object's *category*, *type* and *group*.
  - Call *setQuotation ()*.
- Declare a pointer object, ***c*** of *Quotation* class.
  - Use this pointer to create a dynamic object element. Pass the values **"Writing sets"**, **"Ballpoint pen"**, **"General"** that will set the object's *category*, *type* and *group*.
  - Call *setQuotation ()*.
- For each dynamic object (***a, b, c***).
  - Call *display()*
  - Display the final price details by calling *FinalPrice(...)* .
- Deallocate memory for all the dynamic objects.

Continued...

Sample output screen



```
------------------------------------------------------------------------
         Quotation Records Tracking: Filing - items [ Storage Box ]
------------------------------------------------------------------------

Enter quotation received for Jan: RM 500.00
Enter quotation received for Feb: RM 480.00
Enter quotation received for Mac: RM 300.80
Enter quotation received for Apr: RM 650.00
Enter quotation received for May: RM 455.00

------------------------------------------------------------------------
        Quotation Records Tracking: Pads - items [ Sticky Notes ]
------------------------------------------------------------------------

Enter quotation received for Jan: RM 800.00
Enter quotation received for Feb: RM 790.00
Enter quotation received for Mac: RM 899.00
Enter quotation received for Apr: RM 700.00
Enter quotation received for May: RM 600.00

------------------------------------------------------------------------
   Quotation Records Tracking: Writing sets - items [ Ballpoint pen ]
------------------------------------------------------------------------

Enter quotation received for Jan: RM 860.00
Enter quotation received for Feb: RM 910.00
Enter quotation received for Mac: RM 700.00
Enter quotation received for Apr: RM 688.00
Enter quotation received for May: RM 780.00

------------------------------------------------------------------------
        The Summary: Stationary Purchased 2017 [Kenchana Corporation]
------------------------------------------------------------------------

  Category        Type          Group       Quotation (RM)   Final Price(RM)
  ------         ------       ----------    -------------     -----------
  Filing          Storage Box    Admin         2385.80          2147.22
  Pads            Sticky Notes  Executive      3789.00          3410.10
  Writing sets    Ballpoint pen General        3938.00          3544.20
```

End of Page.